



# Fundamentos de Python

**Lección 3: Instalar y administrar IDEs.  
Mi primer programa en Python.**

# ÍNDICE

<b>Instalar y administrar IDEs. Mi primer programa.....</b>	<b>2</b>
1      Presentación y Objetivos .....	2
2      Principales IDEs para programar en python .....	3
Spyder .....	3
Visual Studio Code .....	4
Pycharm.....	4
PyDev .....	5
IDLE .....	6
3      Instalación Spyder en Ubuntu 16.04.....	7
Instalación Spyder en Windows 10 .....	8
Instalación Visual Studio Code en Ubuntu 16.04.....	13
Instalación Visual Studio Code en Windows 10 .....	15
4      Principales librerías. Mi primer programa.....	22
Numpy .....	22
Matplotlib .....	28
<b>Puntos clave.....</b>	<b>32</b>

# Instalar y administrar IDEs. Mi primer programa.

## 1 PRESENTACIÓN Y OBJETIVOS

En este capítulo mostraremos los principales entornos de desarrollo (IDEs) para desarrollar nuestro código en el lenguaje de programación en Python. Del conjunto de IDEs disponibles, seleccionaremos dos de ellos: **Visual Studio Code** y **Spyder**, y mostraremos cuál es el procedimiento a seguir para instalar y configurar cada uno de estos IDEs en los sistemas operativos Ubuntu 16.04 y Windows 10.

Adicionalmente, haremos un breve repaso sobre las principales librerías utilizadas en Python, haciendo uso de ellas en nuestro primer programa implementado en Python.



### Objetivos

- Conocer los principales IDEs para programar en Python.
- Saber instalar y configurar los IDEs en diferentes sistemas operativos.
- Conocer las principales librerías de Python.
- Saber ejecutar un programa en Python.

## 2 PRINCIPALES IDEs PARA PROGRAMAR EN PYTHON

Un entorno de desarrollo integrado o IDE (del inglés Integrated Development Environment) es una aplicación informática que proporciona un conjunto de servicios para facilitar al programar el desarrollo de software. En la actualidad podemos encontrar una gran variedad de IDEs que pueden facilitarnos la tarea de desarrollar código, y la elección del más adecuado es una tarea crucial, ya que nos facilitará en mayor o menor medida el desarrollo de código, así como la ejecución y depuración de errores de nuestro programa.

De entre el conjunto de IDEs disponibles, en este primer apartado mostraremos los cinco más conocidos, aunque en esta asignatura seleccionaremos dos de ellos.

### Spyder

Es un entorno de desarrollo integrado de código abierto y gratuito para programación científica en el lenguaje Python. La fecha de lanzamiento inicial fue el 18 de Octubre de 2009, y su autor original es Pierre Raybaut.

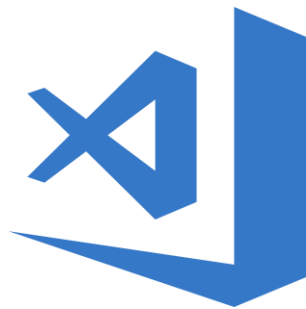


*Figura 2.1: Logo IDE Spyder*

Spyder ofrece un nivel avanzado de edición, depuración y funciones de explorador de datos. Además, es compatible con la consola ipython.

## Visual Studio Code

Es un editor de código desarrollado por Microsoft para Windows, Linux y macOS. Entre sus principales ventajas se destaca el soporte para la depuración, control integrado de Git y finalización inteligente de código entre otros. La fecha de lanzamiento de este IDE fue en abril de 2015.



*Figura 2.1: Logo IDE Visual Studio Code*

Visual Studio Code se puede extender a través de su repositorio. Una característica a remarcar es la capacidad de crear extensiones que analizan código, como linters y herramientas para análisis estático.

## Pycharm

Es uno de los IDE más completos y populares entre los desarrolladores de Python. Incluye funciones inteligentes que facilitan la tarea del programador, como un editor de código con sugerencias, que analiza lo que escribe y te ofrece opciones para ayudarte autocompletar el código que el programador se encuentra desarrollando.



*Figura 2.2: Logo IDE Pycharm*

Fue desarrollado por JetBrains y lanzado por primera vez en febrero de 2010. A diferencia de otros IDEs que hemos analizado en esta lección, Pycharm es un software de pago, aunque cuenta con una versión de prueba gratuita para que los desarrolladores puedan probarlo sin coste.

## **PyDev**

Este entorno de desarrollo es de código abierto, y aunque no incluye tantas funciones como las analizadas anteriormente, cuenta con herramientas útiles, como el autocompletado de código, sangrados inteligentes y un depurador.



*Figura 2.3: Logo IDE PyDev*

En concreto, Pydev es un complemento de terceros de Eclipse. Desarrollado por Appcelerator en julio de 2003.

## IDLE

Es un entorno de desarrollo integrado para Python, incluido de forma predeterminada al instalar Python (desde la versión 1.5.2b1). Creado por el autor original de Python, Guido van Rossum en diciembre de 1998.

A diferencia del resto de IDEs analizados anteriormente, IDLE no cuenta con tantas ventajas, aunque incluye herramientas como el resaltado de sintaxis, sangría inteligente y un depurador con puntos de interrupción persistentes.



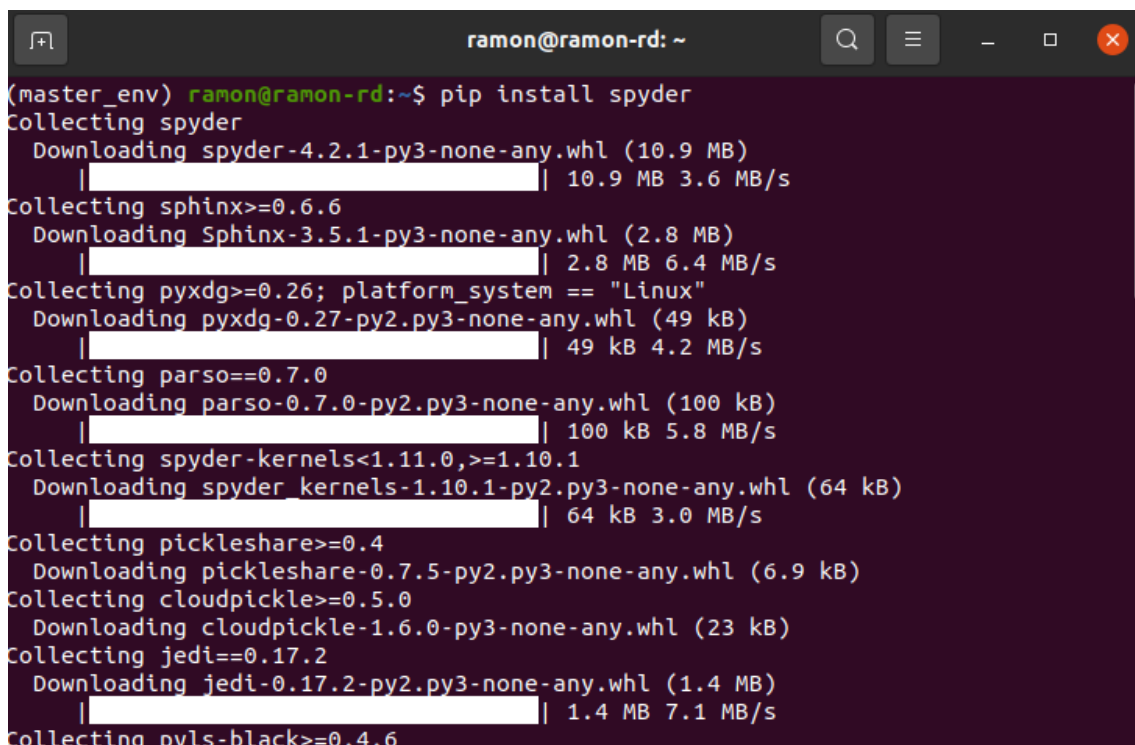
### Importante

**Dado que existe una gran variedad de entornos de desarrollo integrado para desarrollar nuestros programas en Python, para esta asignatura hemos seleccionado los IDEs Spyder y Visual Studio Code. Aunque siempre es posible seleccionar otro IDE de su preferencia.**

### 3 INSTALACIÓN SPYDER EN UBUNTU 16.04

La instalación de este IDE en Ubuntu 16.04 va a ser realizada por medio de comandos en una terminal. Para ello, abrimos una terminal (dirigiéndonos al menú de Ubuntu, o pulsando la combinación de teclas Ctrl+Alt+T) y escribimos la siguiente orden:

**pip install spyder**



```
(master_env) ramon@ramon-rd:~$ pip install spyder
Collecting spyder
  Downloading spyder-4.2.1-py3-none-any.whl (10.9 MB)
    | [Progress Bar] | 10.9 MB 3.6 MB/s
Collecting sphinx>=0.6.6
  Downloading Sphinx-3.5.1-py3-none-any.whl (2.8 MB)
    | [Progress Bar] | 2.8 MB 6.4 MB/s
Collecting pyxdg>=0.26; platform_system == "Linux"
  Downloading pyxdg-0.27-py2.py3-none-any.whl (49 kB)
    | [Progress Bar] | 49 kB 4.2 MB/s
Collecting parso>=0.7.0
  Downloading parso-0.7.0-py2.py3-none-any.whl (100 kB)
    | [Progress Bar] | 100 kB 5.8 MB/s
Collecting spyder-kernels<1.11.0,>=1.10.1
  Downloading spyder_kernels-1.10.1-py2.py3-none-any.whl (64 kB)
    | [Progress Bar] | 64 kB 3.0 MB/s
Collecting pickleshare>=0.4
  Downloading pickleshare-0.7.5-py2.py3-none-any.whl (6.9 kB)
Collecting cloudpickle>=0.5.0
  Downloading cloudpickle-1.6.0-py3-none-any.whl (23 kB)
Collecting jedi>=0.17.2
  Downloading jedi-0.17.2-py2.py3-none-any.whl (1.4 MB)
    | [Progress Bar] | 1.4 MB 7.1 MB/s
Collecting pyls-black>=0.4.6
```

Figura 3.1: Instalación Spyder Ubuntu 16.04

Una vez finalice el proceso, tendremos instalado Spyder en nuestra máquina. Para ejecutarlo, podemos escribir en una terminal el comando “spyder”, o buscar la aplicación en el menú de inicio.



## Instalación Spyder en Windows 10

Accedemos a la web de Spyder para descargar el IDE:

<http://docs.spyder-ide.org/current/installation.html>


### Standalone installers

Our standalone installers for Windows and macOS are available from Spyder 4.2 onwards. We recommend using this installation method on those platforms, but we offer several other options for Linux, advanced users and specific needs, so keep reading if that's the case for you.



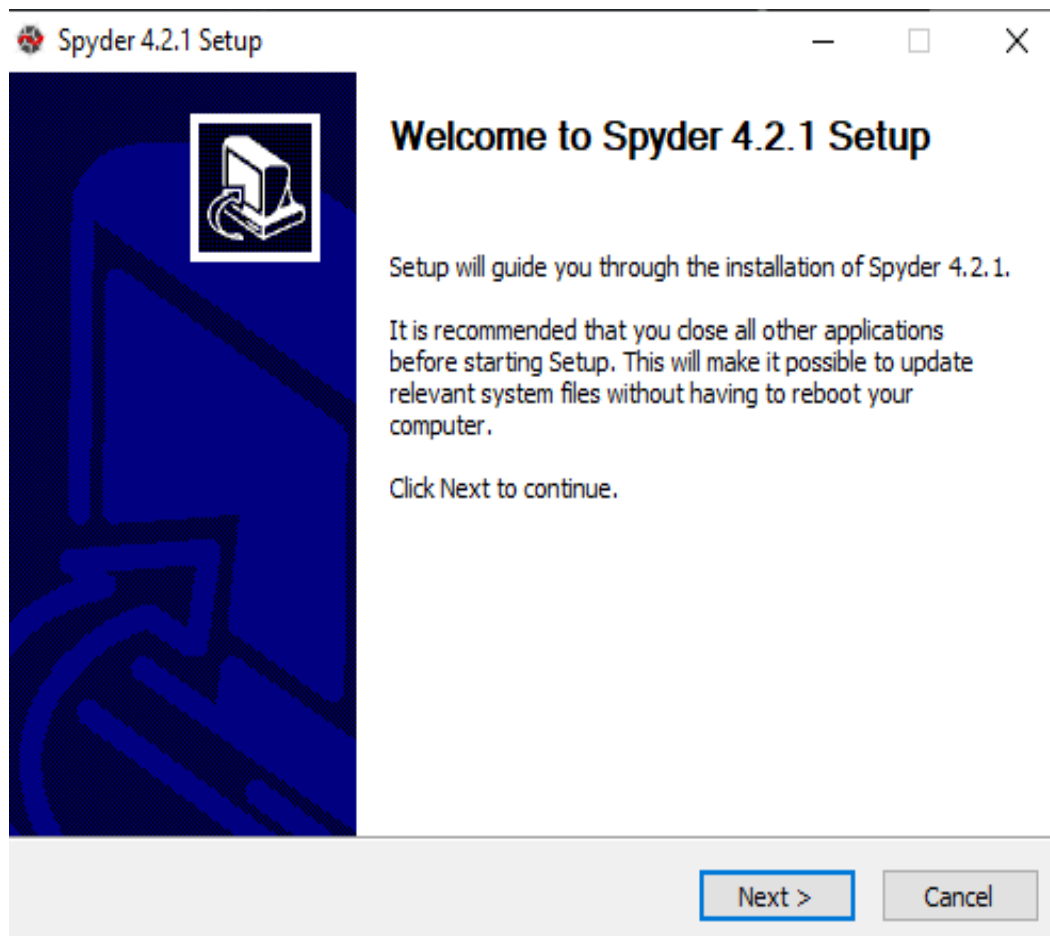
Figura 3.2: Web para descarga de Spyder en Windows 10

Una vez descargado, lo ejecutamos para realizar su instalación. A continuación, mostramos un conjunto de capturas de pantalla para mostrar el proceso de instalación.



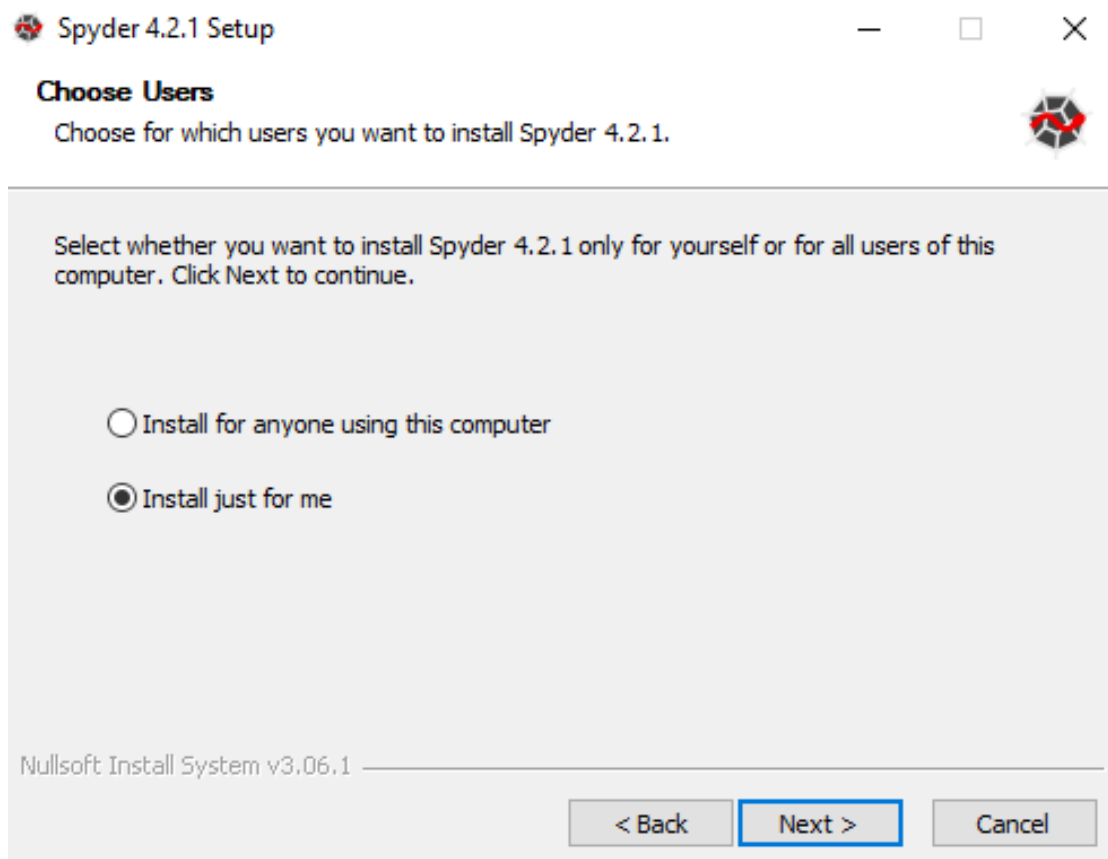
**Importante**

El proceso de instalación de este IDE puede demorarse un poco, tenga paciencia.



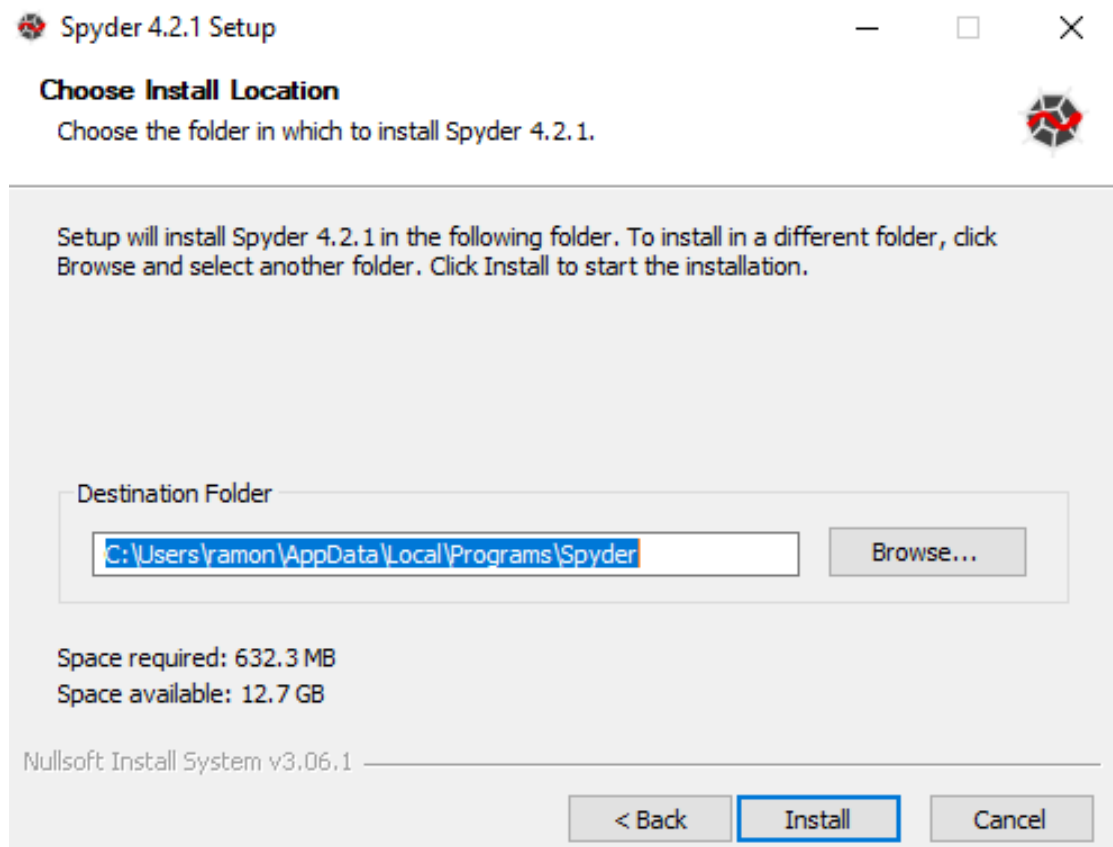
*Figura 3.3: Proceso instalación Spyder en Windows 10 - 1/4*

El instalador ofrece la posibilidad de realizar una instalación del **IDE Spyder** para todos los usuarios de vuestra máquina o solo para el usuario que está activo. En nuestro caso particular, hemos decidido instalar Spyder solo para un usuario, en caso contrario marquen la primera opción.



*Figura 3.4: Proceso instalación Spyder en Windows 10 - 2/4*

A continuación, seleccionamos el directorio donde deseamos realizar la instalación. Podéis seleccionar la opción por defecto o bien elegir una ruta alternativa.



*Figura 3.5: Proceso instalación Spyder en Windows 10 - 4/4*

Tras pulsar siguiente y esperar a que finalice el proceso de instalación, habremos finalizado.

En la siguiente captura se muestra un ejemplo del IDE Spyder, indicando detalladamente cuál es la distribución del entorno, así como un sencillo ejemplo de un programa en el lenguaje de programación Python.

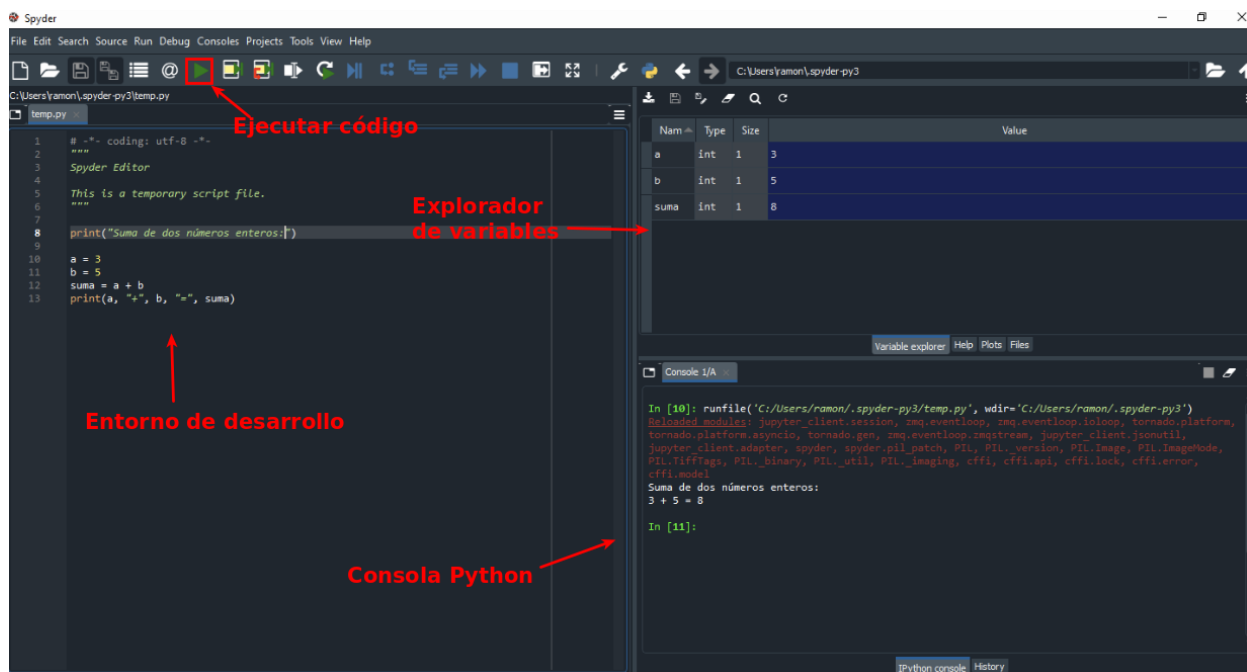


Figura 3.6: Entorno de desarrollo integrado Spyder

## Instalación Visual Studio Code en Ubuntu 16.04

En primer lugar, nos dirigimos a la web oficial de Visual Studio Core: <https://code.visualstudio.com/> y accedemos a la sección de descargas para obtener la versión adecuada.

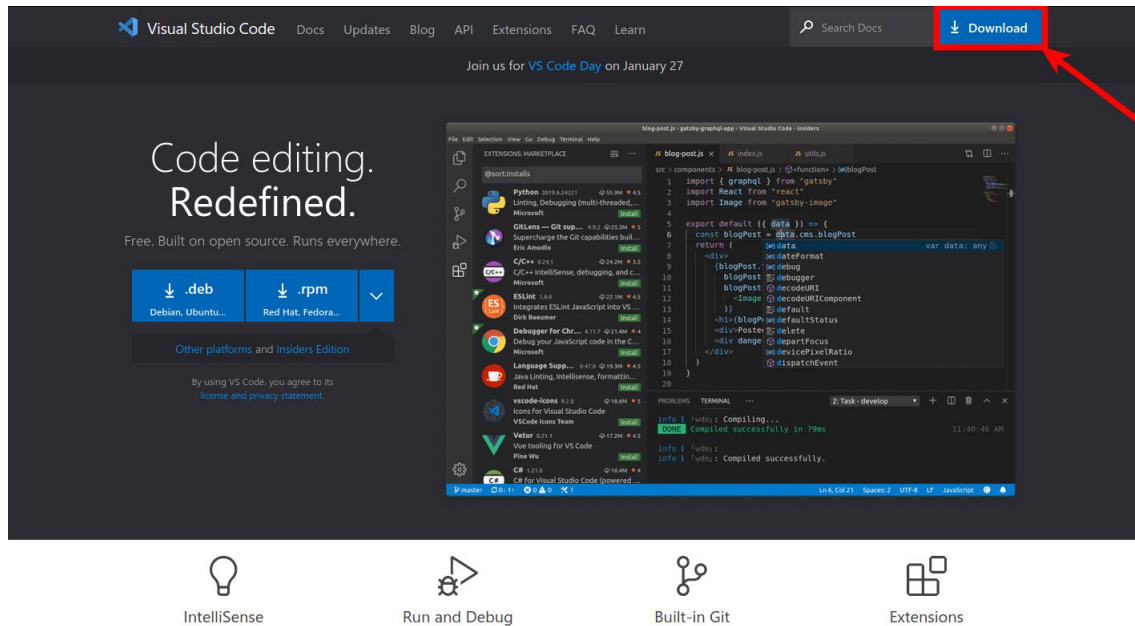


Figura 3.7: Descarga Visual Studio Code en Ubuntu 16.04 - 1/2

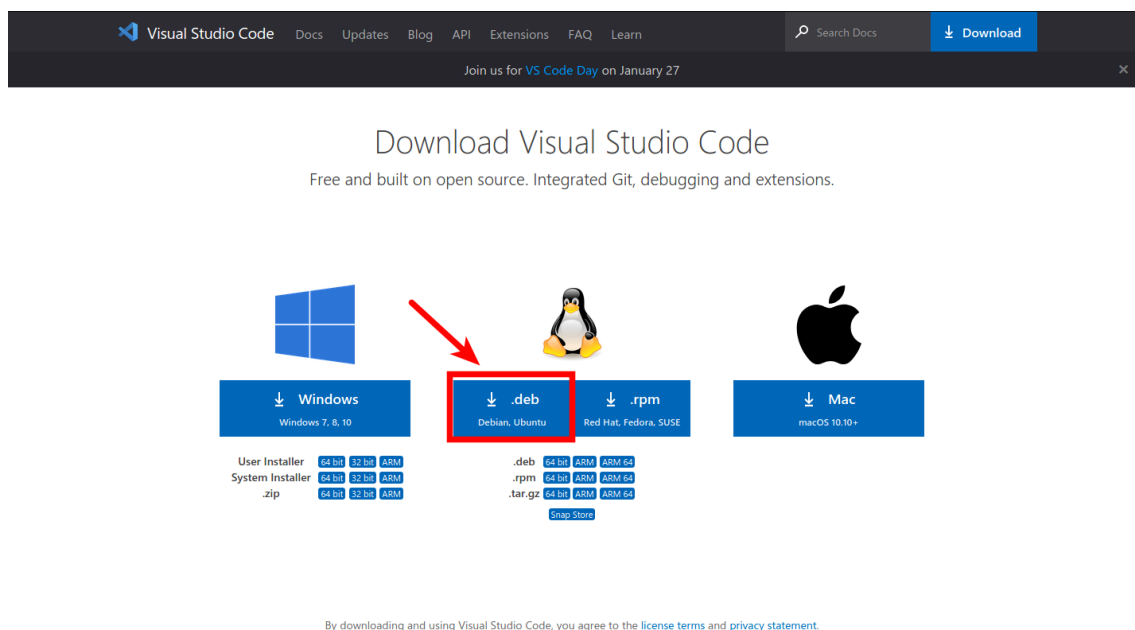
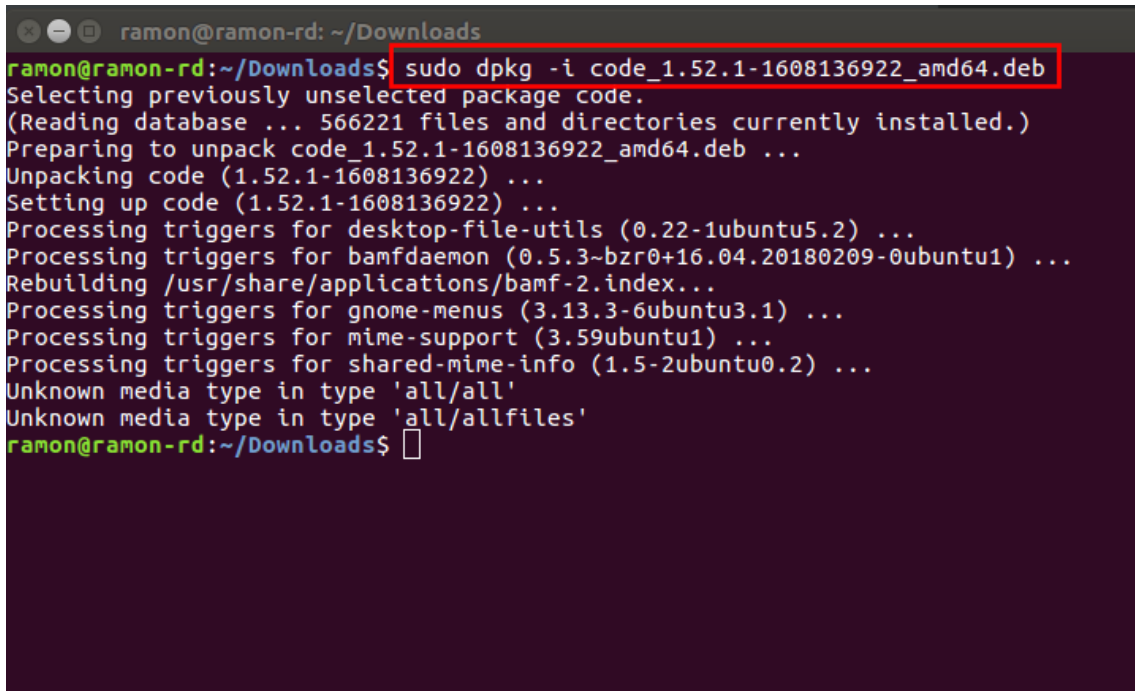


Figura 3.8: Descarga Visual Studio Code en Ubuntu 16.04 - 2/2

Una vez hayamos descargado el instalador, abrimos una terminal, nos dirigimos al directorio donde se ha realizado la descarga (en mi caso, como se encuentra en el directorio Descargas he utilizado la orden "cd Descargas". A continuación, realizamos la instalación.



```
ramon@ramon-rd: ~/Downloads
ramon@ramon-rd:~/Downloads$ sudo dpkg -i code_1.52.1-1608136922_amd64.deb
Selecting previously unselected package code.
(Reading database ... 566221 files and directories currently installed.)
Preparing to unpack code_1.52.1-1608136922_amd64.deb ...
Unpacking code (1.52.1-1608136922) ...
Setting up code (1.52.1-1608136922) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5.2) ...
Processing triggers for bamfdaemon (0.5.3~bzip0+16.04.20180209-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for mime-support (3.59ubuntu1) ...
Processing triggers for shared-mime-info (1.5-2ubuntu0.2) ...
Unknown media type in type 'all/all'
Unknown media type in type 'all/allfiles'
ramon@ramon-rd:~/Downloads$
```

Figura 3.9: Instalación Visual Studio Code Ubuntu 16.04

Una vez finalice el proceso, tendremos instalado el IDE en nuestra máquina.

## Instalación Visual Studio Code en Windows 10

En primer lugar, nos dirigimos a la web oficial de Visual Studio Core: <https://code.visualstudio.com/> y accedemos a la sección de descargas para obtener la versión adecuada.

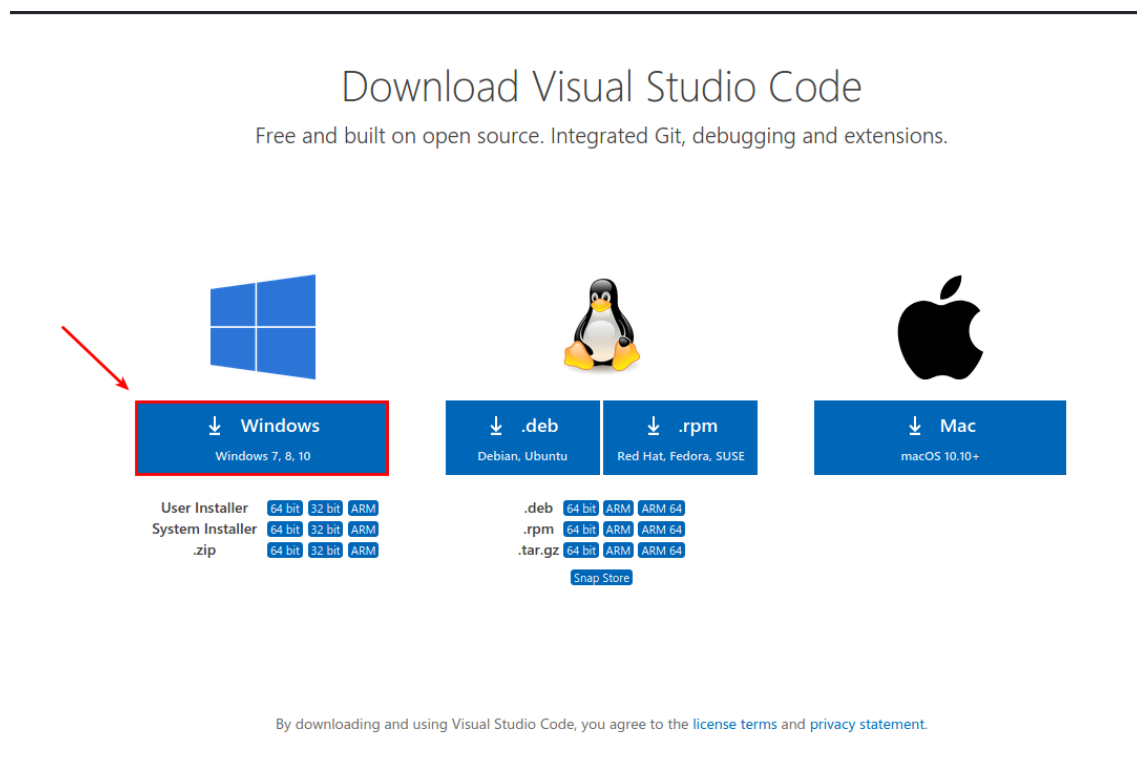


Figura 3.10: Descarga Visual Studio Code en Windows 10

Ejecutamos el instalador que hemos descargado y nos aparecerá una ventana como la de la imagen 16. Tras leer el acuerdo de licencia, aceptamos los términos y pulsamos siguiente.



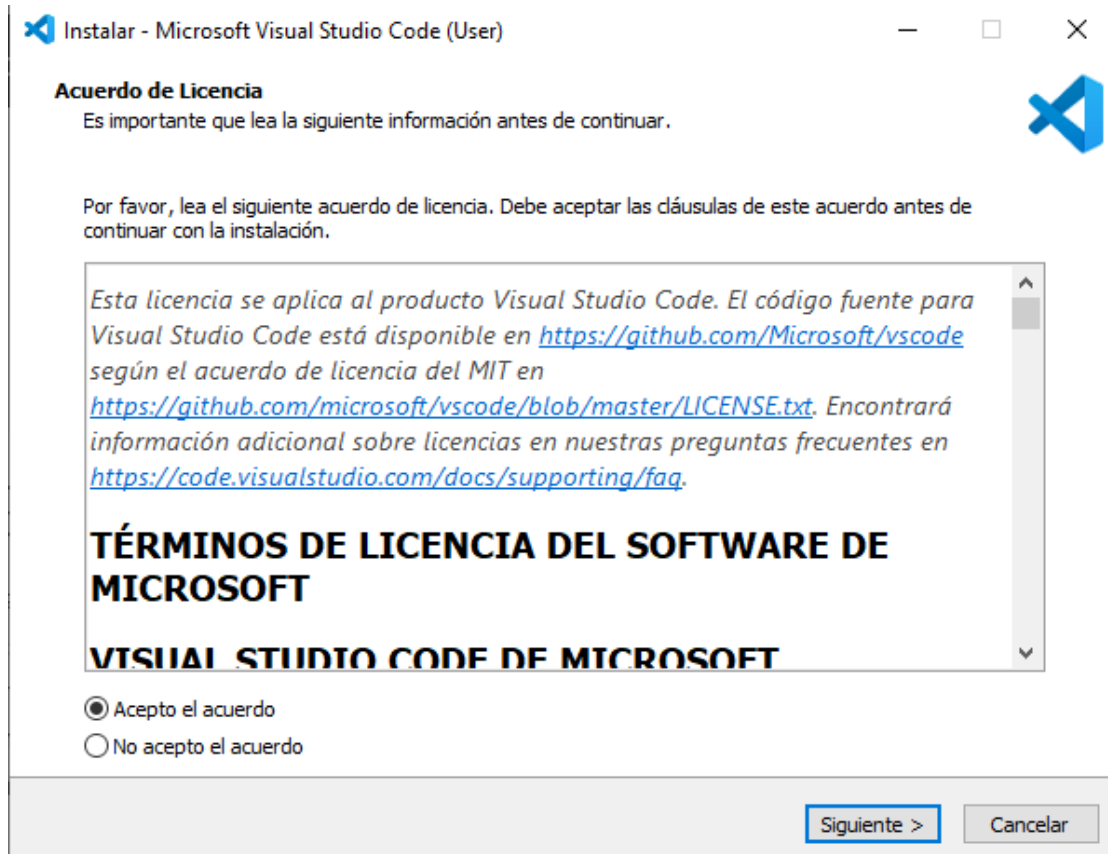


Figura 3.11: Descarga Visual Studio Code en Windows 10

A continuación, seleccionamos la ruta donde queremos realizar la instalación de Visual Studio Code. En este caso práctico, hemos seleccionado la ruta que viene establecida por defecto.

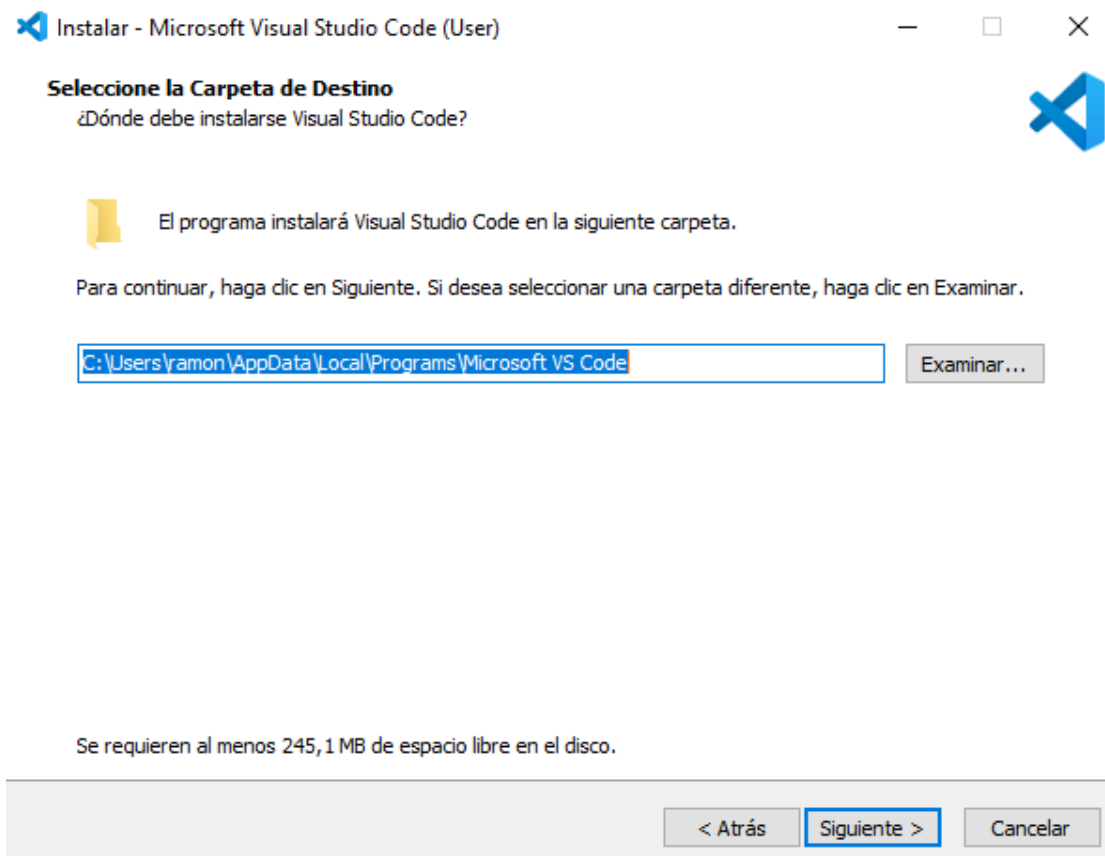
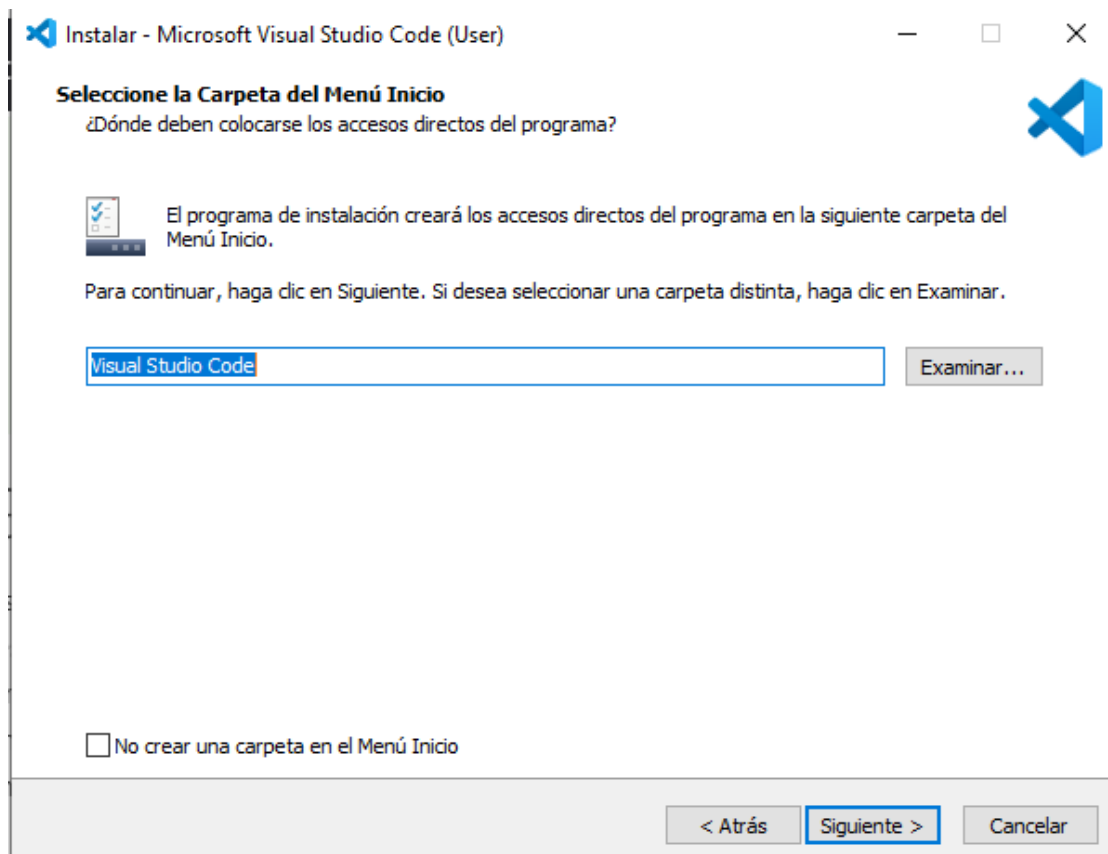


Figura 3.12: Proceso de instalación Visual Studio Code en Windows 10 - 2/6

El instalador también nos permite la posibilidad de elegir el directorio donde deseamos crear el acceso directo del IDE. Por defecto, se incluye el acceso directo en el Menú Inicio.



*Figura 3.13: Proceso de instalación Visual Studio Code en Windows 10 - 3/6*

Casi para terminar, en la siguiente ventana del instalador marcamos la opción "Agregar a PATH". Tras finalizar la instalación, debéis reiniciar vuestras máquinas.

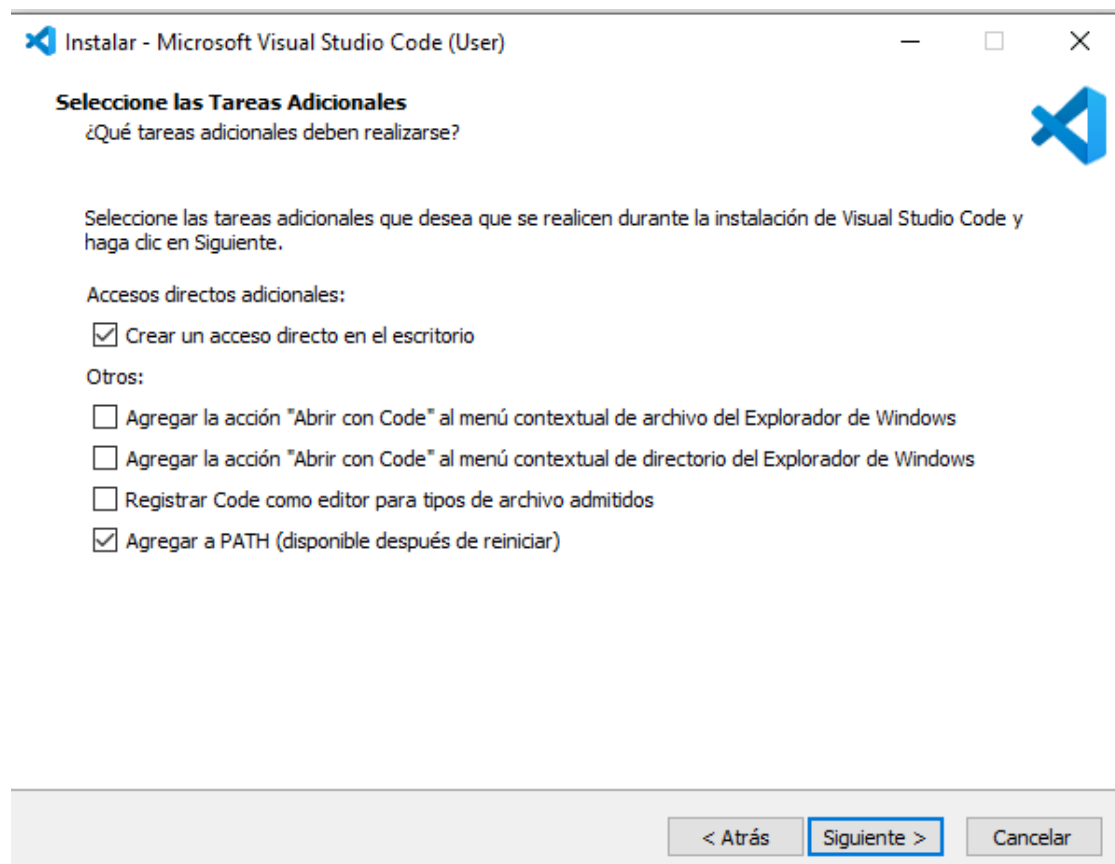


Figura 3.14: Proceso de instalación Visual Studio Code en Windows 10 - 4/6

Finalizamos la instalación de Visual Studio Code pulsando "Instalar". Una vez finalice el proceso de instalación, habremos concluido con la instalación del IDE Visual Studio Code.

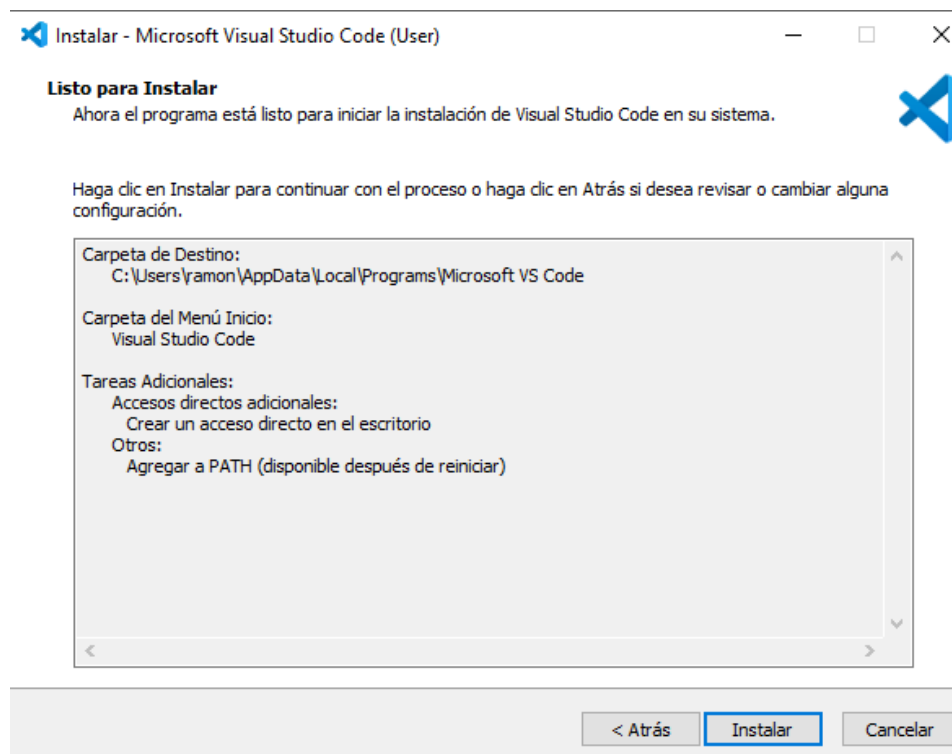


Figura 3.15: Proceso de instalación Visual Studio Code en Windows 10 - 5/6

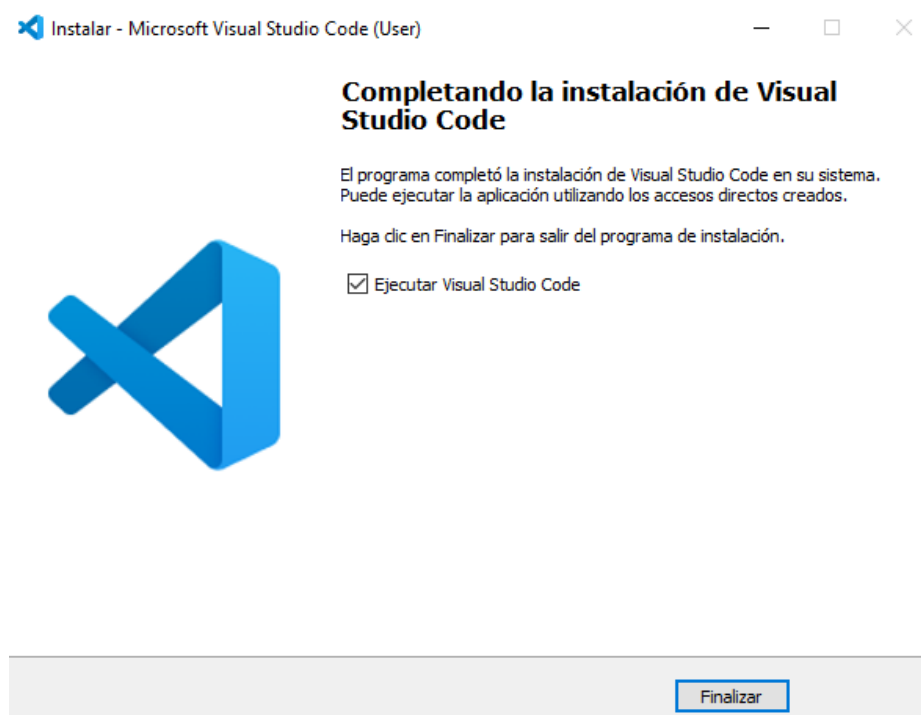


Figura 3.16: Proceso de instalación Visual Studio Code en Windows 10 - 6/6

En la siguiente captura se muestra un ejemplo del IDE Visual Studio Code, indicando cuál es la disposición de su entorno, así como un sencillo ejemplo de un programa en el lenguaje de programación Python

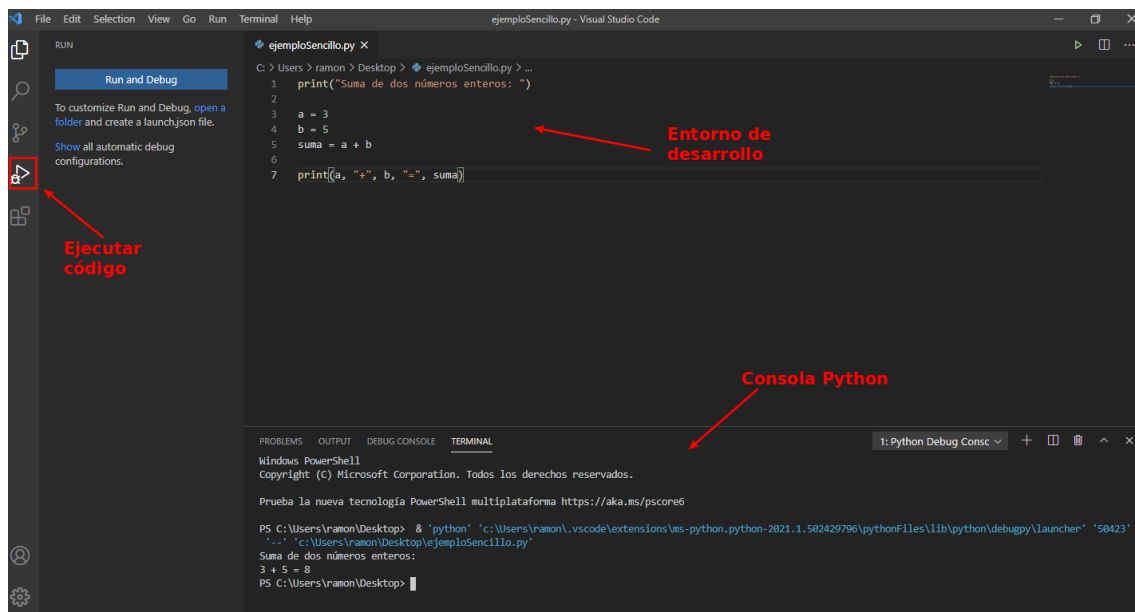


Figura 3.17: Entorno de desarrollo integrado Visual Studio Code

## 4 PRINCIPALES LIBRERÍAS. MI PRIMER PROGRAMA.

Como hemos estudiado en previas lecciones, existen una gran multitud de librerías o paquetes disponibles en Python. Dependiendo del problema que el desarrollador quiera solucionar, podrá seleccionar la que más se adecúe a su tarea. Sin embargo, consideramos que hay dos librerías de Python principales que todo programador debe conocer. Hablamos de **numpy** y **matplotlib**.

### Numpy

Numpy es el paquete fundamental para cálculo científico en Python. Es una biblioteca de Python que proporciona un objeto del tipo matriz multidimensional y una variedad de operaciones rápidas para realizar cálculos sobre matrices, incluyendo operaciones de entrada salida, álgebra lineal, operaciones estadísticas básicas y manejo de números aleatorios entre otros.



#### *Importante*

La librería **numpy** no se instala por defecto al instalar una versión de Python, sino que tenemos que instalarla nosotros manualmente (haciendo uso por ejemplo, del gestor de paquetes **pip** que estudiamos en la lección anterior).

Para instalar numpy en nuestra máquina, podemos hacerlo desde la propia terminal de nuestro IDE favorito. En el caso que se muestra en esta memoria, se hará uso del IDE Spyder. A continuación, se detalla como instalar numpy en Ubuntu 16.04 (si utiliza Windows, siga las instrucciones de la lección 2).

```
(base) ramon@ramon-rd:~$ pip install numpy
Collecting numpy
  Downloading numpy-1.20.1-cp38-cp38-manylinux2010_x86_64.whl (15.4 MB)
    |████████████████████████████████████████| 15.4 MB 4.2 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.20.1
(base) ramon@ramon-rd:~$
```

Figura 4.1: Instalación del paquete numpy en Ubuntu

Una vez finalice el proceso de instalación, tendremos numpy instalado en nuestra máquina. A continuación se muestra el proceso a seguir para crear un programa en Spyder.



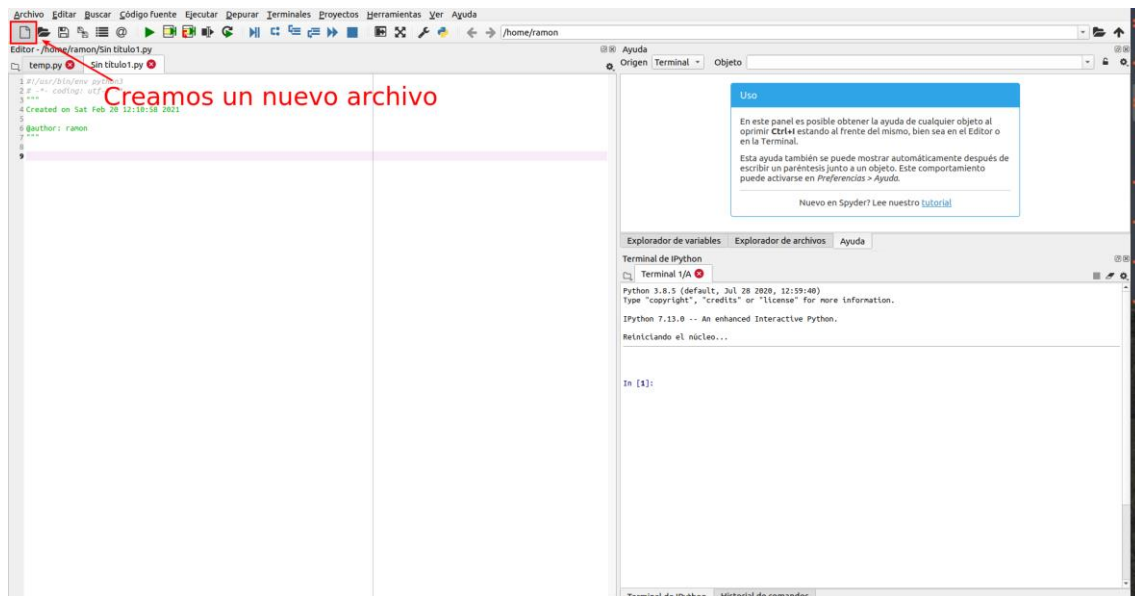


Figura 4.2: Crear un programa nuevo en Spyder

A continuación, implementamos un programa sencillo de prueba. Este código utiliza la librería numpy para crear un array de tamaño 10, cuyos valores son elementos aleatorios en el intervalo [0, 20).

Posteriormente, recorreremos el array y mostramos si los números son pares o impares.

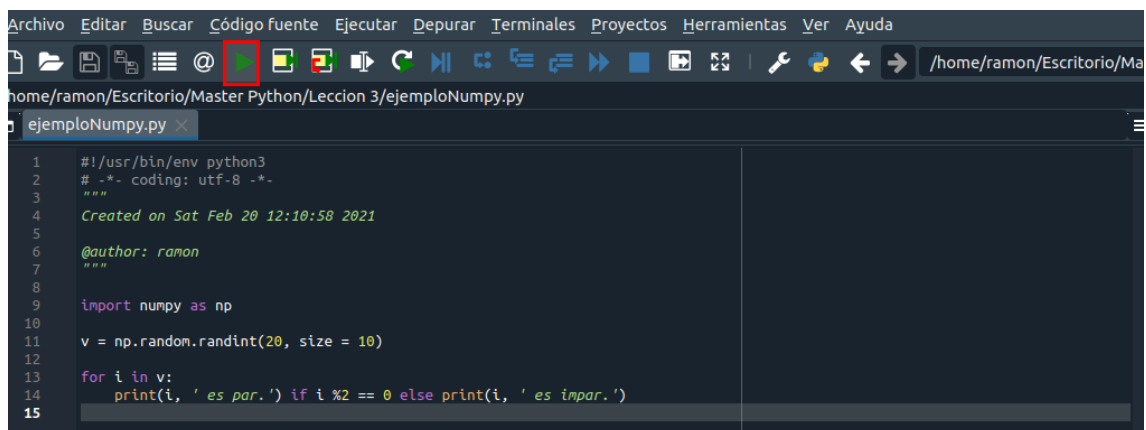
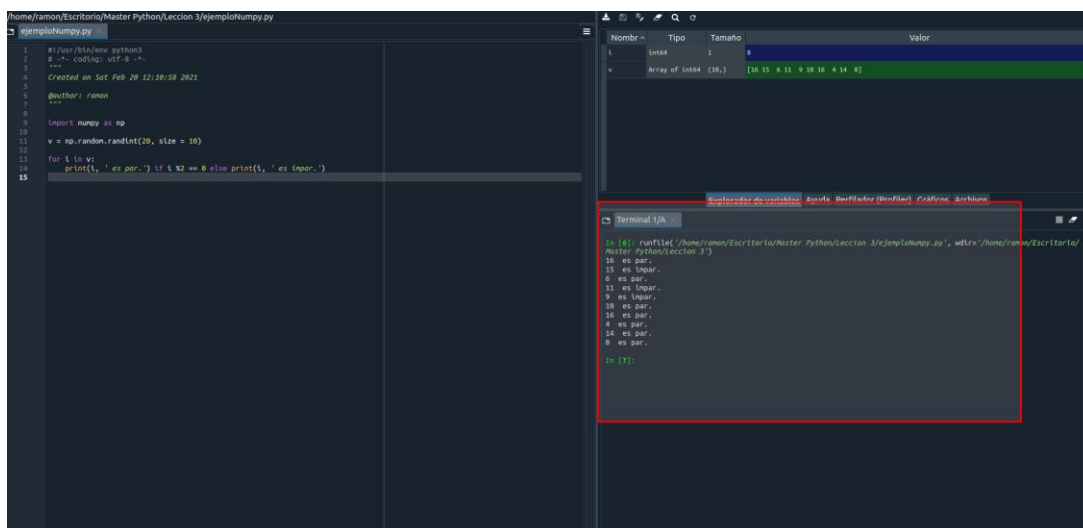


Figura 4.2: Ejecutar un programa en Spyder

Para ejecutar este programa, debemos pulsar el icono “play”, que se encuentra en la parte superior del IDE.

Tras ejecutar nuestro programa de prueba, podemos observar dos cosas interesantes:

1. En la terminal de iPython, ubicada en la esquina inferior derecha, podemos encontrar los resultados obtenidos tras ejecutar nuestro programa de ejemplo:



*Figura 4.3: Visualización de resultados en Spyder*

2. Por otro lado, en la parte superior derecha, podemos encontrar el explorador de variables. En la siguiente captura podemos apreciar dos variables. La primera, la variable *i* contiene el último elemento (8) del array *v*. Por otro lado, se muestra cómo la variable *v* hace referencia a un array de tamaño 10, cuyos valores son números enteros, que fueron generados de forma aleatoria (con la función `randint`).

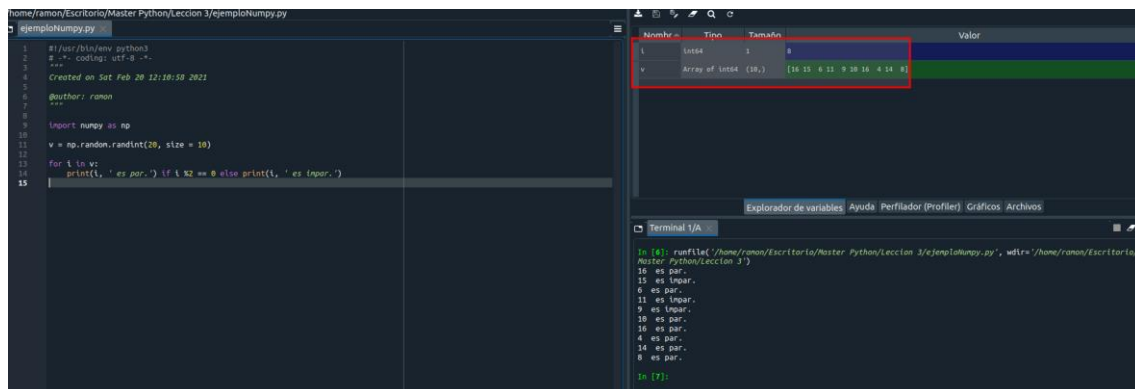


Figura 4.4: Explorador de variables Spyder

Este mismo código puede ser creado o cargado, y ejecutado en el IDE Visual Studio Code. En este caso, tras abrir el programa desarrollado anteriormente, pulsamos el icono "play" ubicado en la parte superior derecha:

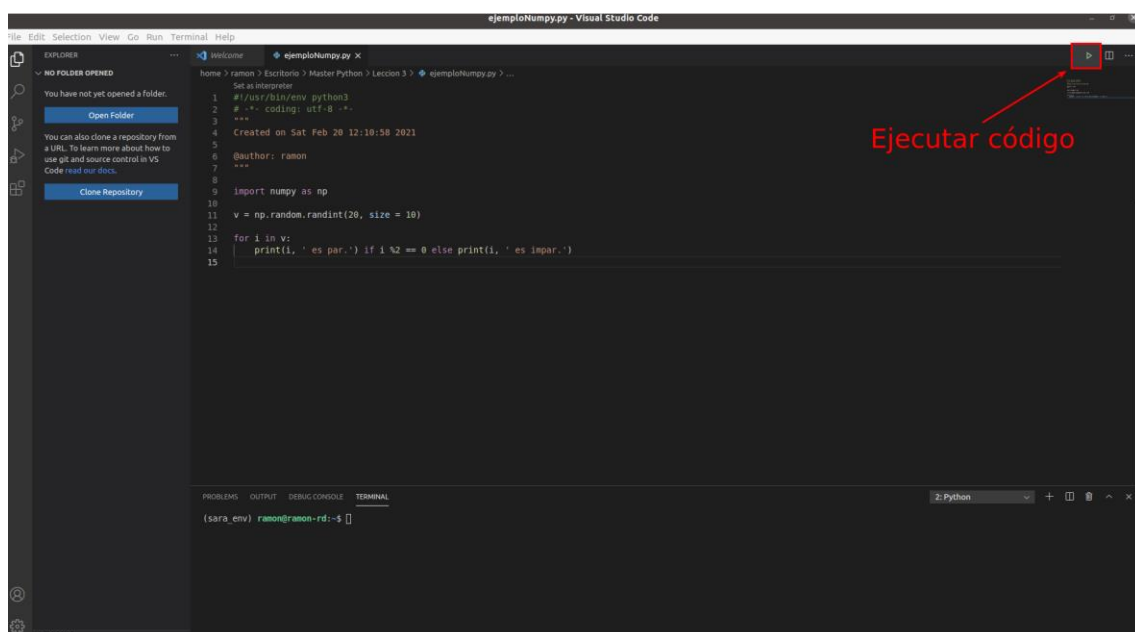
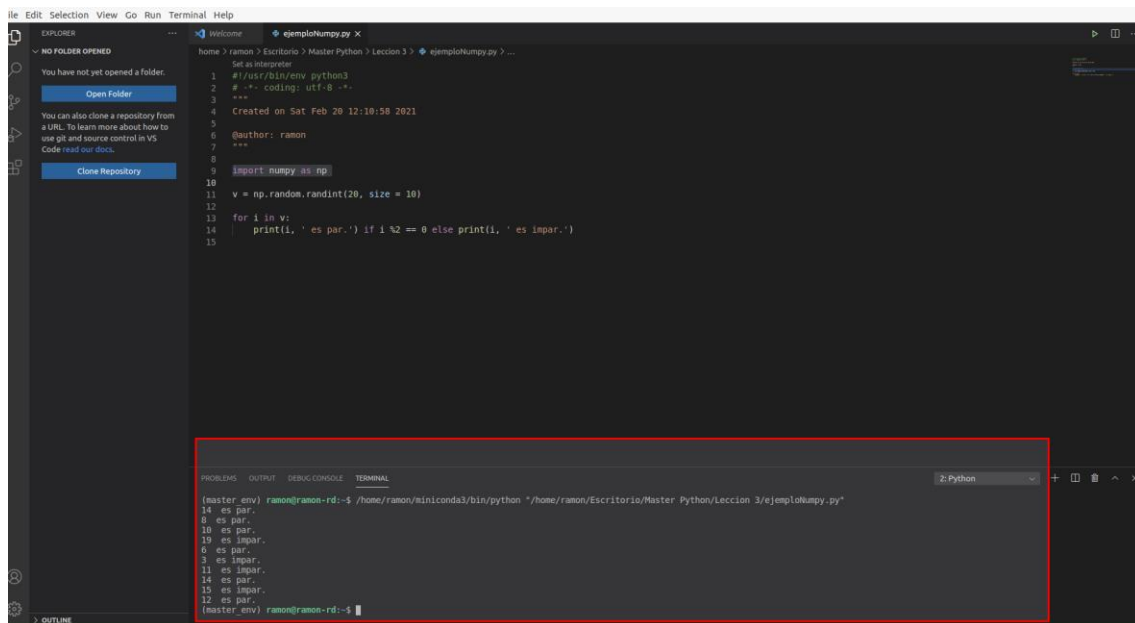


Figura 4.5: Ejecutar código Python en Visual Studio Code

Por último, en la parte inferior del IDE encontramos el resultado tras ejecutar el programa:



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a file named 'ejemploNumpy.py'. The main editor area displays the following Python code:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Feb 20 12:10:58 2021
5
6 @author: ramon
7 """
8
9 import numpy as np
10
11 v = np.random.randint(20, size = 10)
12
13 for i in v:
14     print(i, ' es par.') if i % 2 == 0 else print(i, ' es impar.')
15
```

The TERMINAL panel at the bottom shows the output of the script:

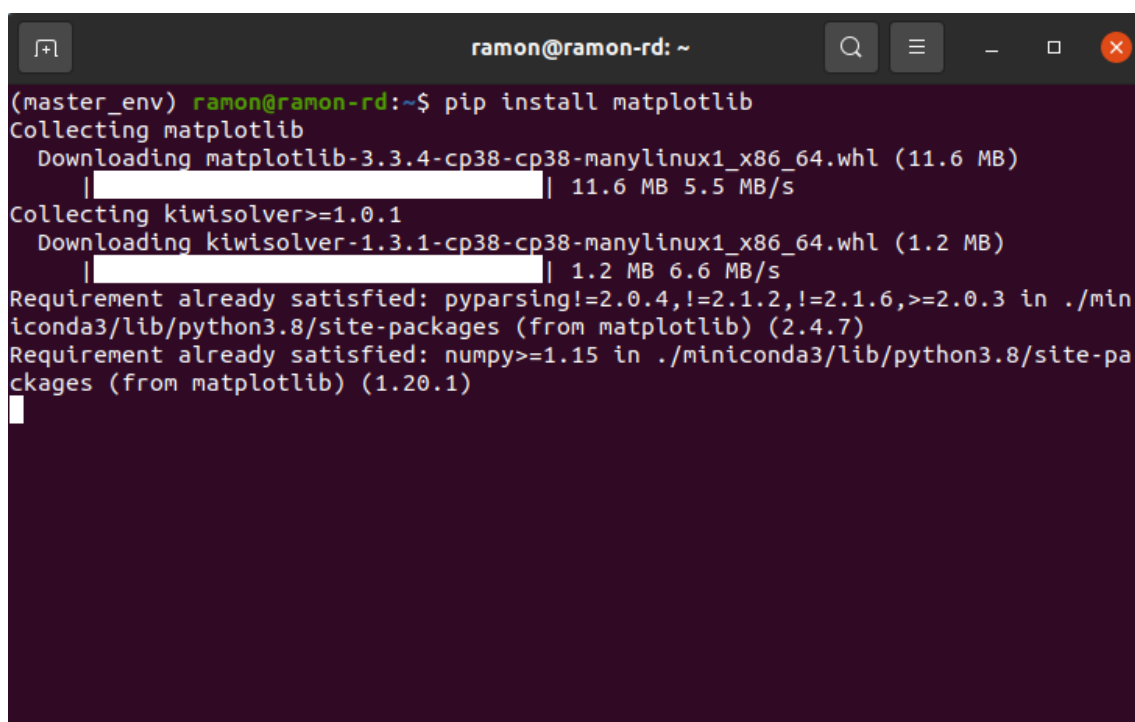
```
(master env) ramon@ramon-rd:~$ /home/ramon/miniconda3/bin/python "/home/ramon/Escritorio/Master Python/Leccion 3/ejemploNumpy.py"
14 es par.
8 es par.
10 es par.
19 es impar.
6 es par.
9 es impar.
11 es impar.
14 es par.
12 es par.
15 es impar.
```

Figura 4.6: Visualización de resultados en Visual Studio Code

## Matplotlib

Matplotlib es una librería desarrollada por John D. Hunter para la generación de gráficos a partir de datos contenidos en arrays en el lenguaje de programación Python y su extensión NumPy.

Para instalar matplotlib en nuestras máquinas, volvemos a hacer uso del gestor de paquetes pip. En concreto, escribimos el siguiente comando en una terminal (en el caso de esta memoria, se está utilizando el sistema operativo Ubuntu 16.04).

A terminal window titled 'ramon@ramon-rd: ~' with search, menu, and window control icons. The terminal shows the command 'pip install matplotlib' being executed. The output indicates that matplotlib is being collected and downloaded (11.6 MB at 5.5 MB/s). It also shows that kiwisolver is being collected and downloaded (1.2 MB at 6.6 MB/s). Finally, it states that the requirements for 'pyparsing' and 'numpy' are already satisfied in the current environment.

```
(master_env) ramon@ramon-rd:~$ pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.3.4-cp38-cp38-manylinux1_x86_64.whl (11.6 MB)
    | _____ | 11.6 MB 5.5 MB/s
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.1-cp38-cp38-manylinux1_x86_64.whl (1.2 MB)
    | _____ | 1.2 MB 6.6 MB/s
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in ./miniconda3/lib/python3.8/site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: numpy>=1.15 in ./miniconda3/lib/python3.8/site-packages (from matplotlib) (1.20.1)
```

Figura 4.7: Instalación del paquete `matplotlib` en Ubuntu

Una vez instalado, procedemos a implementar y ejecutar un ejemplo. En este ejemplo

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Feb 20 13:00:10 2021

@author: ramon
"""
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 34, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]

x = np.arange(len(labels)) # the label locations
width = 0.35 # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

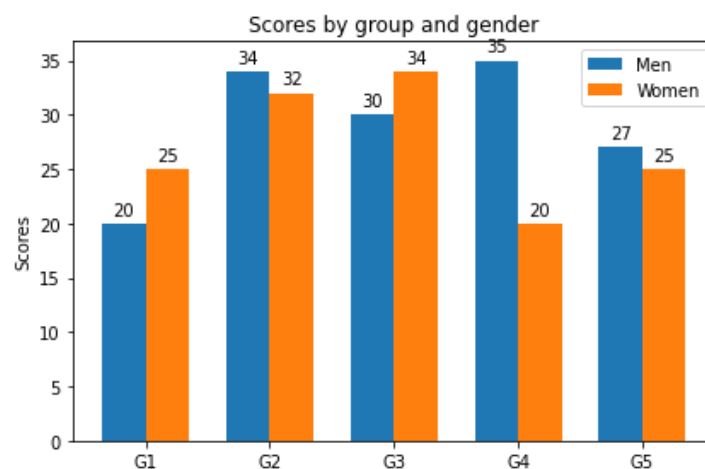
def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3), # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```

Figura 4.8: Ejemplo código Matplotlib 1/2 (código original en el [enlace](#))



```
import numpy as np
import matplotlib.pyplot as plt

category_names = ['Strongly disagree', 'Disagree',
                  'Neither agree nor disagree', 'Agree', 'Strongly agree']

results = {
    'Question 1': [10, 15, 17, 32, 26],
    'Question 2': [26, 22, 29, 10, 13],
    'Question 3': [35, 37, 7, 2, 19],
    'Question 4': [32, 11, 9, 15, 33],
    'Question 5': [21, 29, 5, 5, 40],
    'Question 6': [8, 19, 5, 30, 38]
}

def survey(results, category_names):
    """
    Parameters
    -----
    results : dict
        A mapping from question labels to a list of answers per category.
        It is assumed all lists contain the same number of entries and that
        it matches the length of *category_names*.
    category_names : list of str
        The category labels.
    """
    labels = list(results.keys())
    data = np.array(list(results.values()))
    data_cum = data.cumsum(axis=1)
    category_colors = plt.get_cmap('RdYlGn')(
        np.linspace(0.15, 0.85, data.shape[1]))

    fig, ax = plt.subplots(figsize=(9.2, 5))
    ax.invert_yaxis()
    ax.xaxis.set_visible(False)
    ax.set_xlim(0, np.sum(data, axis=1).max())

    for i, (colname, color) in enumerate(zip(category_names, category_colors)):
        widths = data[:, i]
        starts = data_cum[:, i] - widths
        ax.barh(labels, widths, left=starts, height=0.5,
                label=colname, color=color)
        xcenters = starts + widths / 2

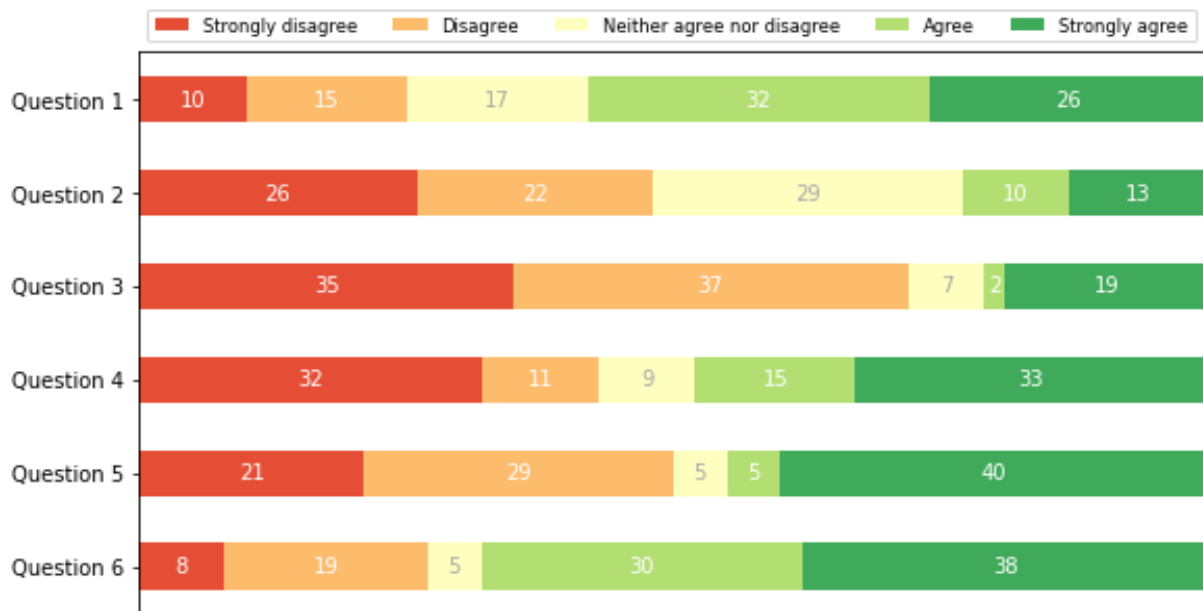
        r, g, b, _ = color
        text_color = 'white' if r * g * b < 0.5 else 'darkgrey'
        for y, (x, c) in enumerate(zip(xcenters, widths)):
            ax.text(x, y, str(int(c)), ha='center', va='center',
                    color=text_color)
    ax.legend(ncol=len(category_names), bbox_to_anchor=(0, 1),
            loc='lower left', fontsize='small')

    return fig, ax

survey(results, category_names)
plt.show()
```

Figura 4.9: Ejemplo código Matplotlib 2/2 (código original en el [enlace](#))

Y el gráfico generado asociado al código anterior es:





## Puntos clave

En esta lección hemos aprendido:

- | Qué es un IDE y cuáles son los más conocidos para desarrollar código en Python.
- | Cómo instalar un IDE en los diferentes sistemas operativos (Windows 10 y Ubuntu 16.04).
- | Saber utilizar los IDEs instalados previamente.
- | Cuáles son las principales librerías de Python.
- | Implementar y ejecutar un programa de prueba haciendo uso de las librerías numpy y matplotlib

